Sistemas Distribuídos 2016/2017

3ª Parte – Guião de Segurança

Professor Naércio Magaia

Turno de 2ª feira, às 11h

Grupo A45 - https://github.com/tecnico-distsys/A45-Komparator

Rafael Belchior
80970

João Trindade
80805

Pedro Gomes,
81534

Lisboa, 5 de maio de 2017

Notas iniciais:

Os LoggingHandler estão configurados como pedido no enunciado: o LoggingHandler está configurado para capturar mensagens, no cliente e no servidor, antes e depois de cada cadeia de processamento de mensagens SOAP, para se poder conferir as alterações efetuadas às mensagens SOAP.

Obtenção do projeto:

mkdir A45
cd A45
git clone https://github.com/tecnico-distsys/A45-Komparator

Instalação do projeto

É necessário ter o UDDI instalado na máquina, assim como a biblioteca de JAX-WS Handlers.

Em Linux: Executar o comando disponibilizado pelos professores. Este comando compila todos os módulos necessários à execução do projeto, ordenadamente.

mvn clean install -DskipITs

Execução do projeto

Em Linux:
Dentro da diretoria supplier-ws, criar duas instâncias de supplier, em terminais diferentes:
mvn exec:java -Dws.i=1
mvn exec:java -Dws.i=2

Dentro de mediator-ws, criar uma instância do mediador: mvn exec:java.

Dentro de mediator-client, criar uma instância do cliente do mediador: mvn exec:java.

Demonstração

A configuração do LoggingHandler está feita de modo a imprimir mensagens SOAP antes e depois da proteção. Destaque dos elementos seguros nas mensagens capturadas: tag “Timestamp” e tag “messageDigest”

Ao se executar o mediator-cliente, este vai correr os testes do buyCart. É possível, através do LoggingHandler, ver as mensagens SOAP antes e depois de serem modificadas pelos handlers.

A nossa aplicação é bem-sucedida se tivermos os certificados guardados, localmente. No entanto, quando mudámos para tentar apanhar os certificados através do CA, alojado na RNL, falha algo.

Deste modo, como não conseguimos obter os certificados, o método getCertificate falha e, consequentemente, a validação da assinatura também falha. Contudo, o handler que trata da verificação está logicamente correto.

A confidencialidade é garantida pela encriptação do número do cartão de crédito, que está a funcionar. A frescura é garantida pelo timestamp, que também está a funcionar. O handler do supplier-client, responsável pela assinatura digital também funciona, até ao método getCertificate.

Demonstração de ataque

Quando o mediador envia uma mensagem ao supplier, esta é passível de ser intercetada e modificada. Este tipo de ataque pode ser prevenido, através da utilização de assinaturas digitais.

Seguindo a sugestão do enunciado, simulámos a alteração não autorizada de mensagem,ao mudar a quantidade de items a comprar. Para isso, criámos um caso de teste, attackTest, dentro do buyProductTest e um handler, AttackHandler. Definimos um id de produto (id “XPTOATTACK”), que despoleta o comportamento de ataque.

O handler AttackHandler interceta as mensagens que chegam ao supplier. Se o id do produto for “XPTOATTACK”, o handler modifica o campo “quantity”, colocando um valor diferente. Deste modo, o ataque é simulado. Quando o supplier receber a mensagem, esta estará adulterada. Ao validar a assinatura digital, ocorrerá um erro.

O handler está bem implementado e funciona. No entanto, não vai passar aos testes, pois não conseguimos obter os certificados. A dependência do handler attackHandler encontra-se comentada no handler-chain do supplier-client, assim como o teste do ataque, attackTest.